UnifiPay Direct API Reference — AI Integration Guide

How to Use This Document

- This document contains the complete specifications for the UnifiPay Direct API. The original source is api-reference-direct/index.html (an HTML page intended for human readers), and this document is a direct transcription of the API specifications contained therein. Last synchronized with the source page on 2026-09-09.
- The AI coding assistant relies solely on this document when writing payment integration code. It does not make assumptions about content not included in this document, or fill in gaps by drawing from other UnifiPay documents (such as the Classic/PG-oriented api-reference/index.html).
- Direct is the "payment link issuance" method. In this workflow, the merchant's server first issues a payment link and shares it with the customer, who then proceeds with the payment directly via that link; this differs from the API configuration used in Classic.
- In the text, terms such as apiKey, apiSecret, X-API-Key, signature, and transactionId are all example values provided for illustrative purposes. When actually integrating the system, these must be replaced with the actual values issued by UnifiPay.
- Sections not found on the original page (e.g., automatic recurring payments) are also omitted from this document. Direct does provide a refund API: file the refund with the Refund API (DIRECT), then the seller signs it with their own Unifi Wallet on the page returned in the response (see section 2.5). Filing from the Refund Management menu in the Unifi Pay Partner Center console remains available as well.
- This page is the latest documentation, based on API v2. The previous documentation remains available for partners using API v1. APIs that exist only on the v2 path are marked in this document with a line reading "This API is available only on the v2 path", wrapped in a pair of HTML-comment marker lines. Fields that exist only in the v2 response are marked with "(v2 only)" at the end of the description.

Table of Contents

Overview

This is an API reference for developers integrating the UnifiPay API. It provides a detailed explanation of how to integrate the entire process - from creating stablecoin payments to settlement - via the REST API. For information on the integration process and business details, refer to the Guides page.

Three Key Features:

Feature	Description
Quick Integration	Complete payment integration with REST API only, no SDK required
HMAC Security	HMAC signature authentication applied to API requests
Webhook Support	Receive real-time payment status via asynchronous Webhook

---

1. API Integration Diagram

This diagram illustrates the data flow between the merchant's server, the customer, and Unifi Pay using the payment link issuance method.

Legend:

- Solid arrow = Synchronous call (Sync)
- Gray dashed arrow (open arrowhead) = Response
- Orange dashed arrow = Asynchronous Webhook (Async)
  Actors: Customer (Browser), Your Server (Backend), Unifi Pay (API Server)

1.1 Payment Diagram

This shows the entire flow from issuing a payment link to receiving a webhook.

1. **[Synchronous]** Your Server -> Unifi Pay: `POST /api/seller/v2/payment/link`
   Headers: `X-API-Key`, `X-Timestamp`, `X-Authorization-Hmac`
2. **[Response]** Unifi Pay -> Your Server: `200 OK` - `linkUrl`, `linkId`
3. **[Synchronous]** Your Server -> Customer: Share the payment link (`linkUrl`) with the customer
4. **[Synchronous]** Customer -> Unifi Pay: Accesses the payment link
   (From this point onward, the process takes place on the Unifi Pay payment link page.)
5. **[Synchronous]** Customer -> Unifi Pay: A payment request is generated when the customer clicks "Proceed to Payment" `transactionId` (UUID), `orderId` = `LINK-{linkId}-{12-character random string}`
6. **[Synchronous]** Customer -> Unifi Pay: Connect wallet and approve payment (on-chain transfer)
7. **[Asynchronous Webhook, 1st]** Unifi Pay -> Your Server: `POST {callbackUrl}`
   Sent only if `callbackUrl` is registered. Body: `status`: `PAID`
8. **[Asynchronous Webhook, 2nd]** Unifi Pay -> Your Server: `POST {callbackUrl}`
   Sent only if `callbackUrl` is registered. Body: `status`: `CONFIRMED` / `FAILED` `CANCELED`
   is not part of this two-step sequence: it occurs while the payment is still in `CREATED`, so it never follows a `PAID` Webhook. Do not assume a fixed delivery count; branch on the `status` value you receive.

Additional Explanation:

- **Your Server**: Requests a payment link from Unifi Pay using the `apiKey` and HMAC signature.
- **Payment Link**: Share the issued `linkUrl` with the customer. There are no restrictions on the sharing channel.
- **Customer**: When the customer accesses the link and proceeds with the payment, a payment request is generated, and the payment is completed via wallet connection and signature.
- **Webhook**: Sent only if a `callbackUrl` was registered when the payment link was issued. The payment result is sent asynchronously twice: the 1st at `PAID` and the 2nd at `CONFIRMED`. This is the only channel through which the payment result is delivered, so it must be registered.

Why `callbackUrl` is effectively mandatory: The Payment Status API (section 2.3) is keyed by `transactionId`, and `transactionId` is issued only at the moment the customer proceeds with the payment on the link page. It is not returned in the payment link issuance response (section 2.2 returns only `linkUrl` and `linkId`). The Webhook is therefore the only channel through which the merchant server is notified of a `transactionId`. Register `callbackUrl` even though the field is marked Optional in the request table.

If `callbackUrl` was not registered, the transactions of a link can still be enumerated after the fact with the section 2.10 Payment Link Transaction API, which is keyed by `linkId` rather than `transactionId`. That is a polling path, not a notification path, so it does not replace the Webhook for receiving results.

---

2. API Detail Reference

Base URL

Environment	Base URL
`Preview`	https://app-api-pay.unifi.me
`Production`	https://api-pay.unifi.me

2.1 Authentication

HMAC authentication is required by default for API requests between servers. The signature value must be included in the `X-Authorization-Hmac` header.

Default Authentication Rule: Unless otherwise specified, HMAC authentication is the default for all server-to-server APIs. APIs that use a different authentication method (e.g., unauthenticated public APIs) are noted separately for each API.

HMAC Generation Formula

```
BASE64(HMACSHA256(apiSecret, {HTTP_METHOD}{URI}{X-API-Key}{X-Timestamp}
{REQUEST_BODY}))
```

Example 1 - With a REQUEST_BODY

Endpoint: `POST /api/seller/v2/payment/link`

Request:

```
request header:
X-API-Key: "b6ef2f7c-3e74-438c-b456-689e821992be"
X-Timestamp: 1773017787000
X-Authorization-Hmac: "some-hmac-signature"

request body:
{
  "requestId": "REQ-20260304-0001",
  "storeId": "123",
  "serviceName": "MyShop",
  "itemName": "ITEM-GAME-001",
  "itemPrice": 12.34,
  "orderCurrencyCode": "USD",
  "returnUrl": "https://myshop.com/payment/returnUrl",
  "callbackUrl": "https://myshop.com/payment/callbackUrl",
  "expiresAt": "2026-08-31T00:00:00Z"
}
```

Generate signature:

```
X-Authorization-Hmac = Base64(
  HmacSHA256(
    apiSecret,
    "POST"                                      <- HTTP_METHOD
    + "/api/seller/v2/payment/link"             <- URI
    + "b6ef2f7c-3e74-438c-b456-689e821992be"  <- X-API-Key
    + "1773017787000"                           <- X-Timestamp
    +
"{"requestId":"REQ-20260304-0001","storeId":"123","serviceName":"MyShop","itemName":"
ITEM-GAME-001","itemPrice":12.34,"orderCurrencyCode":"USD","returnUrl":"https://
myshop.com/payment/returnUrl","callbackUrl":"https://myshop.com/payment/
callbackUrl","expiresAt":"2026-08-31T00:00:00Z"}"       <- REQUEST_BODY (no
whitespace)
  )
)
```

Example 2 - Without a REQUEST_BODY

Endpoint: `GET /api/seller/v2/payment/123`

Request:

```
request header:
X-API-Key: "b6ef2f7c-3e74-438c-b456-689e821992be"
X-Timestamp: 1773017787000
X-Authorization-Hmac: "some-hmac-signature"
```

Generate signature:

```
X-Authorization-Hmac = Base64(
  HmacSHA256(
    apiSecret,
    "GET"                                       <- HTTP_METHOD
    + "/api/seller/v2/payment/123"              <- URI
    + "b6ef2f7c-3e74-438c-b456-689e821992be"    <- X-API-Key
    + "1773017787000"                           <- X-Timestamp
                                                <- REQUEST_BODY omitted (none)
  )
)
```

Example 3 - With Query Parameters

Endpoint: `GET /api/seller/v2/payment/settlement/transaction`

Request:

```
request url:
GET /api/seller/v2/payment/settlement/transaction?
from=2026-03-01T00:00:00Z&to=2026-03-31T23:59:59Z&paymentType=PURCHASE&page=0&size=20

request header:
X-API-Key: "b6ef2f7c-3e74-438c-b456-689e821992be"
X-Timestamp: 1773017787000
X-Authorization-Hmac: "some-hmac-signature"
```

Generate signature:

```
X-Authorization-Hmac = Base64(
  HmacSHA256(
    apiSecret,
    "GET"                                       <- HTTP_METHOD
    + "/api/seller/v2/payment/settlement/transaction"  <- URI (query string excluded)
    + "b6ef2f7c-3e74-438c-b456-689e821992be"    <- X-API-Key
    + "1773017787000"                           <- X-Timestamp
                                                <- REQUEST_BODY omitted (none)
```

```
  )
)
```

Handling Query Parameters: The `{URI}` used for the signature includes only the path; the query string (everything after `?`) is not included. Signing with the query string included will cause authentication to fail.

Request Header Parameters

Parameter	Type	Required	Description
X-Authorization-Hmac	STRING	Required	The generated HMAC signature value. Required for all server-to-server API requests.
X-API-Key	STRING	Required	The apiKey issued by Unifi Pay.
X-Timestamp	STRING	Required	Unix timestamp in milliseconds. Requests will be rejected if the timestamp differs from the current time by 5 minutes or more.

Timestamp Validity Range: The `X-Timestamp` must be within 5 minutes of the current server time. If it falls outside this range, the request will be automatically rejected to prevent replay attacks.

Server Timestamp Lookup

Retrieve the timestamp from the Unifi Pay server to verify the server's status, and use the timestamp value from the response in the HMAC signature.

```
GET /api/v2/server/time
```

Authentication Method: This API is an unauthenticated public API. Call it without the `X-API-Key`, `X-Timestamp`, and `X-Authorization-Hmac` headers. Since it is used to obtain the timestamp for the HMAC signature, it is excluded from the default authentication rule (HMAC required) described above.

Response Fields:

Field	Type	Description
timestamp	INTEGER(INT64)	The server's timestamp in milliseconds

Success (200 OK):

```
{
  "timestamp": 1773017787000
}
```

Error (500 Internal Server Error):

```
{
  "code": "INTERNAL_SERVER_ERROR",
  "message": ""
}
```

2.2 Create Payment Link API

Issues a payment link for a fixed amount and item. Share the returned link with your customer to open the payment page without any additional integration.

```
POST /api/seller/v2/payment/link
```

Request Body Parameters

Parameter	Type	Length	Required	Description
`requestId`	STRING	64	Required	A unique identifier (idempotency key) for the issuance request. Use only letters, numbers, hyphens, and underscores. Resubmitting the request with the same requestId will be rejected as a duplicate issuance.
`storeId`	STRING	128	Required	Merchant ID information provided by the requester.
`serviceName`	STRING	40	Required	Service or merchant information to be displayed on the link screen and payment window.
`itemName`	STRING	100	Required	Name of the product being sold (product name).
`itemPrice`	DOUBLE	-	Required	Price of the product being sold. Expressed in the order currency and must be greater than 0.
`orderCurrencyCode`	STRING	8	Required	Order currency code following ISO 4217. Supports USD or JPY.
`returnUrl`	STRING	2048	Optional	The URL to which the user will be redirected after payment is complete. If not specified, the payment will proceed without a return button.
`callbackUrl`	STRING	2048	Optional	Webhook URL for receiving the payment result. With the payment link method, no transaction identifier is generated until the

Parameter	Type	Length	Required	Description
				payment occurs, so the result cannot be checked via status inquiry - be sure to specify it. See "Why callbackUrl is effectively mandatory" in section 1.1.
`expiresAt`	STRING	-	Required	The expiration time of the payment link. Must be in ISO-8601 format and represent a future date and time. If the link is accessed after expiration, an expiration error will be returned.

Response Fields

Field	Type	Description
`linkUrl`	STRING	The URL of the generated payment link. Share this with the customer.
`linkId`	STRING	Link identifier. Keep it on the merchant side. To expire a link, use the force-expire button in the payment link list on the console. You can also expire it with the section 2.9 Force Expire Payment Link API.

Example Request:

```
{
  "requestId": "REQ-20260304-0001",
  "storeId": "123",
  "serviceName": "MyShop",
  "itemName": "ITEM-GAME-001",
  "itemPrice": 12.34,
  "orderCurrencyCode": "USD",
  "returnUrl": "https://myshop.com/payment/returnUrl",
  "callbackUrl": "https://myshop.com/payment/callbackUrl",
  "expiresAt": "2026-08-31T00:00:00Z"
}
```

Example Response:

```
{
  "linkUrl": "https://unifipay.example.com/pay/links/3f1d0a549c1e4f6a8a5b2f",
  "linkId": "3f1d0a549c1e4f6a8a5b2f"
}
```

Note: If you share the `linkUrl` from the response with the customer, the payment will proceed without any additional integration. Since `requestId` is an idempotency key,

resubmitting a request with the same value will result in the duplicate issuance being rejected.

Note: This response does not contain `transactionId`. The `transactionId` is issued when the customer proceeds with the payment on the link page and is delivered to the merchant server only through the Webhook (section 2.4).

API version of payment links: A payment is created not when the link is issued but when the customer proceeds with payment on the link, and its API version follows the version used to issue the link.

Error Responses

HTTP Status	Error Code	Description
400	BAD_REQUEST	Invalid input values (format error, missing required values, expiresAt set to a past date, etc.)
401	UNAUTHORIZED	Invalid HMAC
403	FORBIDDEN	Unauthorized IP or Path
503	MAINTENANCE	Under maintenance
406	PAYMENT_INVALID_REQUEST	If the request is resubmitted with the same requestId, or if the callbackUrl is not on the allowlist
406	PAYMENT_UNSUPPORTED_POLICY	Unsupported order currency or amount policy (orderCurrencyCode, itemPrice)
406	PAYMENT_STORE_INACTIVE	When the requested storeId is not in the ACTIVE state
406	PAYMENT_WALLET_NOT_REGISTERED	If the settlement wallet address is not registered
500	INTERNAL_SERVER_ERROR	Internal server error

2.3 Payment Status API

Checks the current status of a created payment.

The response fields and example below are based on the v2 response. Fields that exist only in the v2 response are marked with "(v2 only)" at the end of the description.

```
GET /api/seller/v2/payment/{transactionId}
```

Response Fields

Field	Type	Description
`transactionId`	STRING	Transaction number generated by Unifi Pay.
`orderId`	STRING	Order id generated by Unifi Pay. For transactions through a payment link, it is issued in the LINK-{linkId}-{12-character random string} format.
`storeId`	STRING	Merchant ID information provided by the requester.
`serviceName`	STRING	Service or merchant information to be displayed on the payment page.
`paymentType`	STRING	Payment: PURCHASE / Refund: REFUND (v2 only)
`status`	STRING	See the enum below.
`failType`	STRING	Set when status is FAILED. See the enum below.
`items`	Array	Payment item list. Maximum of 1 item.
`items[].name`	STRING	Identification information for the item.
`items[].price`	DOUBLE	Price of the item.
`countryCode`	STRING	Country code following ISO Alpha-3 (e.g., KOR).
`orderCurrencyCode`	STRING	Order currency code following ISO 4217 (USD, JPY).
`orderAmount`	DOUBLE	Ordered amount.
`payCurrencyCode`	STRING	Type of stablecoin used for payment.
`payAmount`	DOUBLE	Paid amount.
`buyerWalletAddress`	STRING	Buyer wallet address. Null before signing (CREATED).
`netAmount`	DOUBLE	Net amount the partner actually receives, in the payment currency. Provided from CONFIRMED onward. (v2 only)
`blockchainTxId`	STRING	Blockchain transaction hash. Returned after the transaction is confirmed on the blockchain (CONFIRMED), or when it fails (FAILED) after being propagated to the blockchain network.
`blockchainNetworkFee`	DOUBLE	Blockchain network fee. The amount of fee actually consumed is returned after the transaction is confirmed on the blockchain (CONFIRMED), or upon failure (FAILED) after network propagation. (Fully covered by Unifi, so there is no cost to the user.)
`createdAt`	INSTANT	Order payment request datetime (v2 only)
`capturedAt`	INSTANT	Time when the on-chain event was observed. Null before observation. A value here does not mean the payment succeeded, so determine success from status. (v2 only)
`finalizedAt`	INSTANT	Time when the payment ended (confirmed, failed, or canceled). Null before it ends. (v2 only)

Note on the response at `PAID`: `blockchainTxId` and `blockchainNetworkFee` are not returned at `PAID`. The payment amount (`payAmount`) is finalized at `CONFIRMED`, so use the values retrieved after `CONFIRMED` for amount and blockchain information.

`status` enum:

Value	Description
`CREATED`	Payment created
`PENDING`	Payment in progress
`PAID`	Payment successful
`CONFIRMED`	Settlement complete (final)
`FAILED`	Payment failed (final)
`CANCELED`	Payment cancelled (final)

Note on mixing v1 and v2: The `PAID` status appears only when the payment is created with v2 and retrieved via the v2 API. If either the creation or the retrieval is on v1, the payment is reported as `CONFIRMED` without `PAID`, the same as before.

For Direct, a payment is created not when the link is issued but when the customer proceeds with payment on the link, and its API version follows the version used to issue the link.

`failType` enum (set when status is FAILED):

Value	Description
`WALLET_PURCHASE_BLOCKCHAIN_FAILED`	Blockchain failure during transfer from unifi wallet
`WALLET_PURCHASE_INSUFFICIENT_BALANCE`	Insufficient balance in user wallet
`PAYMENT_PURCHASE_BLACKLISTED`	User wallet is in payment blacklist
`PAYMENT_STORE_INACTIVE`	The store was inactive at the time of payment
`WALLET_SESSION_EXPIRED`	Unifi account session validation failed at the payment request stage
`WALLET_TRANSFER_IN_PROGRESS`	A withdrawal or transfer is already in progress for the same wallet and token, so the duplicate request was rejected
`UNKNOWN`	Unknown error during payment

Example Response:

```
{
  "transactionId": "<uuid>",
```

```
  "orderId": "<orderId>",
  "storeId": "<storeId>",
  "serviceName": "<serviceName>",
  "paymentType": "PURCHASE",
  "items": [
    {
      "name": "ITEM-GAME-001",
      "price": 50000
    }
  ],
  "status": "CONFIRMED",
  "failType": null,
  "countryCode": "KOR",
  "payAmount": 49000,
  "netAmount": 48510,
  "payCurrencyCode": "USDT",
  "orderAmount": 50000,
  "orderCurrencyCode": "USD",
  "buyerWalletAddress": "0x742d35Cc6634C0532925a3b844Bc454e4438f44e",
  "blockchainTxId":
"0x4fce0d72ff7f4b9f4ba0cf58d30119924bea3811d85a830c762e1d87b3e1ee17",
  "blockchainNetworkFee": 0.00253778,
  "createdAt": "2026-09-07T04:00:00.000Z",
  "capturedAt": "2026-09-07T04:01:28.000Z",
  "finalizedAt": "2026-09-07T04:01:30.000Z"
}
```


Status Information:

- CONFIRMED: The payment has been confirmed and the settlement is complete (final).
- FAILED: The payment has failed.
  Error Responses

HTTP Status	Error Code	Description
400	BAD_REQUEST	Invalid input value
401	UNAUTHORIZED	If HMAC is invalid
403	FORBIDDEN	If IP or Path is not allowed
503	MAINTENANCE	If under maintenance
406	PAYMENT_TRANSACTION_NOT_FOUND	Invalid transactionId
500	INTERNAL_SERVER_ERROR	Internal server error

2.4 Payment Webhook

Receives the processing result for the payment asynchronously. On a successful payment the Webhook is sent at two points: the 1st at `PAID` and the 2nd at `CONFIRMED`. Both are sent with the same `transactionId`, so handle them idempotently in case of duplicate delivery. You can switch to the payment completion screen once the 1st (`PAID`) Webhook is received. Note that the Webhook is sent only when `callbackUrl` was registered at payment link issuance, and that it is the only path for receiving payment results.

```
POST <merchant callbackUrl>
```

Webhook Body Fields

Field	Type	Required	Description
`transactionId`	STRING	Required	Transaction number generated by Unifi Pay.
`orderId`	STRING	Required	The order ID generated by Unifi Pay. For transactions made via a payment link, this is issued in the format LINK-{linkId}-{12-character random string} and is determined when the customer completes the payment via the link.
`status`	STRING	Required	Payment processing result. PAID: payment successful (1st Webhook). CONFIRMED: settlement complete (2nd Webhook). FAILED: failure. CANCELED: canceled because the transaction remained in the CREATED state for roughly 30 minutes or more, or because the customer left the payment window.
`type`	STRING	Required	Transaction type: PURCHASE or REFUND. Refunds are notified through the same Webhook, in which case transactionId and orderId are the values of the refund transaction (orderId inherits the orderId of the original payment, so it is the same value as the purchase). This field carries the same value as the paymentType field in query responses.

Example Request - 1st Webhook (PAID):

```
{
  "transactionId": "<uuid>",
  "orderId": "LINK-3f1d0a549c1e4f6a8a5b2f-a1b2c3d4e5f6",
  "status": "PAID",
  "type": "PURCHASE"
}
```

Example Request - 2nd Webhook (CONFIRMED):

```
{
  "transactionId": "<uuid>",
  "orderId": "LINK-3f1d0a549c1e4f6a8a5b2f-a1b2c3d4e5f6",
  "status": "CONFIRMED",
  "type": "PURCHASE"
}
```

Important Response Note: The merchant server must return an HTTP 200 OK response code after receiving the webhook.

Detailed Information: Webhooks deliver only minimal information. To retrieve full payment details, call the section 2.3 Payment Status API after receiving the Webhook.

Response at `PAID`: `blockchainTxId` and `blockchainNetworkFee` are not returned at `PAID`. The payment amount (`payAmount`) is finalized at `CONFIRMED`, so use the values retrieved after `CONFIRMED` for amount and blockchain information.

Retry Policy

If the merchant server returns a 5xx error or does not respond, the Webhook is resent according to the policy below.

Order	Retry timing
1st retry	After 3 seconds
2nd retry	After 30 seconds
3rd retry	After 10 minutes
4th retry	After 1 hour
5th retry	After 3 hours

2.5 Refund API (DIRECT)

Refunds can be filed through the seller API. Filing and lookup are offered on both the v1 and v2 paths; this document describes the v2 path.

How it works: File the refund with the refund filing API below, then the seller must sign it directly with their own Unifi Wallet on the `startPageUrl` page returned in the response (billing is not involved). Filing from the Refund Management menu in the Unifi Pay Partner Center console remains available as well.

Note: A refund can only proceed if the original payment is in `CONFIRMED` status.

Filing conditions: In addition to the status condition above, only payments received as DIRECT payout (immediate distribution to the partner wallet) can be refunded. Only one

refund may be in progress per original payment, and a filed refund that is never signed is canceled automatically and excluded from that count.

Refund status values: A DIRECT refund has three states: `PAID` (right after the seller signs), `CONFIRMED` (on-chain collection complete), and `FAILED`. There is no `PENDING` stage.

2.5.1 Refund Filing API

Files a refund for an original payment. The refund proceeds only after the seller signs it on the signing page returned in the response.

```
POST /api/seller/v2/payment/refund/direct
```

Request Body Parameters

Parameter	Type	Length	Required	Description
`requestId`	STRING	128	Required	Refund request identifier. It must be globally unique within the partner.
`orderId`	STRING	128	Required	Order ID of the original payment. Use the orderId from the refund pre-check API response below as is.
`orderCurrencyCode`	STRING	8	Required	Order currency code. It must match the original payment.
`orderAmount`	DOUBLE	-	Required	Refund amount. It must be greater than 0.
`isPartial`	BOOLEAN	-	Required	Whether it is a partial refund. true: partial refund, false: full refund.

Response Fields

Field	Type	Required	Description
`transactionId`	STRING	Required	ID of the created refund transaction. Track its status with the single refund lookup API below.
`startPageUrl`	STRING	Required	URL of the seller signing page. The refund proceeds only after signing on this page.

Example Request:

```
{
  "requestId": "RFD-20260907-0001",
  "orderId": "LINK-3kQ9xR2mN7pLw8vBc1dEfH-a1B2c3D4e5F6",
  "orderCurrencyCode": "USD",
  "orderAmount": 30.0,
```

```
  "isPartial": true
}
```

Example Response:

```
{
  "transactionId": "660e8400-e29b-41d4-a716-446655440111",
  "startPageUrl": "https://unifipay.example.com/refund/start?token=<token>"
}
```

Error Responses

HTTP Status	Error Code	Description
400	BAD_REQUEST	Invalid input value
401	UNAUTHORIZED	Invalid HMAC
403	FORBIDDEN	Unauthorized IP or Path
406	PAYMENT_TRANSACTION_NOT_FOUND	The original payment was not found
406	PAYMENT_INVALID_REQUEST	The original payment is not a DIRECT payout or is not CONFIRMED, a refund is already in progress for the same original payment, the remaining refundable amount is exceeded, the requestId is duplicated, or a currency or amount policy is violated
406	PAYMENT_REFUND_BLACKLISTED	The buyer wallet is blocked from refunds
500	INTERNAL_SERVER_ERROR	Internal server error
503	MAINTENANCE	Under maintenance

2.5.2 Refund Pre-check API

Checks whether a refund is possible and how much is still refundable before filing. It is for guidance only and does not guarantee acceptance, so handle a failed filing as part of the normal flow.

```
GET /api/seller/v2/payment/refund/lookup
```

What to sign: The URI used for the HMAC signature includes the path only, not the query string. See section 2.1 Authentication for details.

Request Query Parameters

Parameter	Type	Required	Description
`transactionId`	STRING	Required	Transaction ID of the original payment.

Response Fields (200 OK)

Field	Type	Required	Description
`transactionId`	STRING	Required	Transaction ID of the original payment.
`orderId`	STRING	Required	Order ID of the original payment. Use it as is for the orderId of the refund filing API.
`status`	STRING	Required	Status of the original payment (public representation).
`orderAmount`	DOUBLE	Required	Order amount of the original payment.
`refundedAmount`	DOUBLE	Required	Sum of confirmed and in-progress refunds. Unsigned drafts are excluded.
`refundableAmount`	DOUBLE	Required	Remaining refundable amount, calculated as orderAmount minus refundedAmount.
`refundable`	BOOLEAN	Required	Pre-check result indicating whether a refund can be filed right now.
`reason`	STRING	Optional	Reason code for why filing is not possible. Null when filing is possible.

`reason` values:

Value	Description
`ORIGINAL_NOT_CONFIRMED`	The original payment is not confirmed yet
`ORIGINAL_NOT_REFUNDABLE`	It ended as failed or canceled
`FULLY_REFUNDED`	It is fully refunded
`NOT_DIRECT_PAYOUT`	It is not a DIRECT payout

Example Response:


```
{
  "transactionId": "550e8400-e29b-41d4-a716-446655440000",
  "orderId": "LINK-3kQ9xR2mN7pLw8vBc1dEfH-a1B2c3D4e5F6",
  "status": "CONFIRMED",
  "orderAmount": 100.0,
  "refundedAmount": 30.0,
  "refundableAmount": 70.0,
  "refundable": true,
```

```
  "reason": null
}
```

Error Responses

HTTP Status	Error Code	Description
401	UNAUTHORIZED	Invalid HMAC
403	FORBIDDEN	Unauthorized IP or Path
500	INTERNAL_SERVER_ERROR	Internal server error
503	MAINTENANCE	Under maintenance

2.5.3 Single Refund Lookup API

Looks up the status of one filed refund. The `transactionId` path variable is not the original payment ID but the refund transaction ID returned by the refund filing response.

```
GET /api/seller/v2/payment/refund/{transactionId}
```

What to sign: The URI used for the HMAC signature includes the path only, not the query string. See section 2.1 Authentication for details.

Response Fields (200 OK)

Field	Type	Required	Description
`transactionId`	STRING	Required	Refund transaction ID.
`orderId`	STRING	Required	Order ID of the original payment.
`status`	STRING	Required	Refund status (public representation). One of PAID, CONFIRMED, or FAILED.
`failType`	STRING	Optional	Reason the refund failed.
`orderAmount`	DOUBLE	Required	Order amount of the refund.
`orderCurrencyCode`	STRING	Required	Order currency code.
`refundAmount`	DOUBLE	Required	Amount actually refunded.
`refundCurrencyCode`	STRING	Required	Currency actually used for the refund.
`blockchainTxId`	STRING	Optional	Blockchain transaction hash. Returned after the transaction is confirmed on the blockchain (CONFIRMED), or when it fails (FAILED) after being propagated to the blockchain network.
`blockchainNetworkFee`	DOUBLE	Optional	Blockchain network fee. The amount of fee actually consumed is returned after the transaction is confirmed on the blockchain

Field	Type	Required	Description
			(CONFIRMED), or upon failure (FAILED) after network propagation. (Fully covered by Unifi, so there is no cost to the user.)

Example Response:


```
{
  "transactionId": "660e8400-e29b-41d4-a716-446655440111",
  "orderId": "LINK-3kQ9xR2mN7pLw8vBc1dEfH-a1B2c3D4e5F6",
  "status": "CONFIRMED",
  "failType": null,
  "orderAmount": 30.0,
  "orderCurrencyCode": "USD",
  "refundAmount": 30.0,
  "refundCurrencyCode": "USDT",
  "blockchainTxId": "0xdef456...",
  "blockchainNetworkFee": 0.0018
}
```


Error Responses

HTTP Status	Error Code	Description
401	UNAUTHORIZED	Invalid HMAC
403	FORBIDDEN	Unauthorized IP or Path
500	INTERNAL_SERVER_ERROR	Internal server error
503	MAINTENANCE	Under maintenance

Integration Notes

Item	Description
Refund webhook	A notification with type set to REFUND is sent to the same callbackUrl as the payment result Webhook. In that case transactionId and orderId are the values of the refund transaction, and orderId inherits the value of the original payment.
Retrieving refunds	Query the section 2.6 Settlement Transaction API below with paymentType=REFUND, or use the section 2.11 Payment Transaction API.
Linking to the original payment	The originTransactionId of a refund points to the transactionId of the original payment.

Note: For refund policies such as the refund wallet, refund amount basis, and address and network conditions, see "5. Refund (Transfer Support Feature)" in the user guide (../guides/index.html#refund).

2.6 Settlement Transaction API

Check payment and refund transaction history.

```
GET /api/seller/v2/payment/settlement/transaction
```

Request Query Parameters

Parameter	Type	Required	Description
from	STRING	Conditional	Start date (based on capturedAt, ISO 8601 UTC. e.g. 2026-03-01T00:00:00Z). Required if transactionId/settlementId not specified; must be earlier than to.
to	STRING	Conditional	End date (based on capturedAt, ISO 8601 UTC. e.g. 2026-03-31T23:59:59Z). Required if transactionId/settlementId not specified; maximum 90-day range allowed.
`paymentType`	STRING	Optional	Payment type filter (PURCHASE / REFUND). All types if not specified.
`transactionId`	STRING	Optional	Transaction ID generated by Unifi Pay. Required if from/to/settlementId not specified.
`settlementId`	STRING	Optional	Unifi Pay settlement ID. Required if from/to/transactionId not specified.
page	INTEGER	Optional	Page number (starting from 0, default: 0)
size	INTEGER	Optional	Page size (default 50, from 1 to 100)

Response Fields (200 OK)

Field Name	Type	Required	Description
`content`	List	Required	List of payment data
`content[].partnerCorpName`	STRING	Required	Corporation name (EN)
`content[].storeId`	STRING	Optional	Store ID
`content[].orderId`	STRING	Required	Order ID for the transaction
`content[].itemName`	STRING	Required	Product identification information
`content[].serviceName`	STRING	Optional	Service or merchant name (EN)
`content[].orderCountryCode`	STRING	Required	Buyer country code

Field Name	Type	Required	Description
`content[].orderCurrencyCode`	STRING	Required	Order currency code following ISO 4217
`content[].orderAmount`	DOUBLE	Required	Order amount
`content[].createdAt`	INSTANT	Required	Order payment request datetime
`content[].transactionId`	STRING	Required	Transaction number generated by Unifi Pay.
`content[].blockchainTxId`	STRING	Required	KaiaScan TxID
`content[].paymentType`	STRING	Required	Payment: PURCHASE / Refund: REFUND
`content[].originTransactionId`	STRING	Optional	Original transaction ID for refunds (required for REFUND, null for PURCHASE)
`content[].status`	STRING	Required	Fixed as CONFIRMED (only CONFIRMED records returned)
`content[].finalizedAt`	INSTANT	Required	Order payment final status datetime
`content[].capturedAt`	INSTANT	Required	Order payment capture datetime
`content[].failType`	STRING	Optional	Order payment failure reason
`content[].payCurrencyCode`	STRING	Required	Actual payment currency USDT
`content[].payAmount`	DOUBLE	Required	Actual payment amount
`content[].netAmount`	DOUBLE	Optional	Net amount the partner actually receives, in the payment currency. Null for refund transactions, which have no receipt. (v2 only)
`content[].paymentExchangeRate`	DOUBLE	Required	Exchange rate at the time of payment. Applied when the payment currency differs from the order currency; 1 if they are the same. (Base currency: payCurrencyCode)
`content[].buyerWalletAddress`	STRING	Required	Buyer's payment wallet address
`content[].variableFeeRate`	DOUBLE	Optional	Payment fee rate (%). Null if settlement receipt not yet generated.
`content[].protocolFeeRate`	DOUBLE	Required	Protocol fee rate as a percentage, where 1 means 1%. This is the rate used to deduct netAmount, and it is currently always 1 for

Field Name	Type	Required	Description
			sellers (MERCHANT_LITE). (v2 only)
`content[].settlementId`	STRING	Optional	Unifi Pay settlement ID. Null if not yet generated.
`content[].settlementCurrencyCode`	STRING	Optional	Settlement currency code (USDT, JPYC). Null if not yet settled.
`content[].settlementExchangeRate`	DOUBLE	Optional	Exchange rate at the time of settlement. Null if not yet settled.
`totalElements`	LONG	Required	Total number of records
`totalPages`	INTEGER	Required	Total number of pages
`pageNumber`	INTEGER	Required	Current page number
`first`	BOOLEAN	Required	Whether this is the first page
`last`	BOOLEAN	Required	Whether this is the last page

Example Response - Success:


```
{
  "content": [
    {
      "partnerCorpName": "ABC Corporation",
      "storeId": "store_001",
      "orderId": "ORD-20260301-0001",
      "itemName": "Premium Subscription",
      "serviceName": "ABC Service",
      "orderCountryCode": "KOR",
      "orderCurrencyCode": "USD",
      "orderAmount": 50000,
      "createdAt": "2026-03-01T10:00:00Z",
      "transactionId": "txn_abc123def456",
      "blockchainTxId": "0xabcdef1234567890abcdef1234567890abcdef12",
      "paymentType": "PURCHASE",
      "originTransactionId": null,
      "status": "CONFIRMED",
      "finalizedAt": "2026-03-01T10:01:30Z",
      "capturedAt": "2026-03-01T10:01:25Z",
      "failType": null,
      "payCurrencyCode": "USDT",
      "payAmount": 10.87,
      "netAmount": 10.76,
      "paymentExchangeRate": 1.0,
```

```
      "buyerWalletAddress": "0x1234567890abcdef1234567890abcdef12345678",
      "variableFeeRate": 0,
      "protocolFeeRate": 1,
      "settlementId": "stl_xyz789",
      "settlementCurrencyCode": "USDT",
      "settlementExchangeRate": 1.0
    }
  ],
  "totalElements": 1,
  "totalPages": 1,
  "pageNumber": 0,
  "first": true,
  "last": true
}
```


Example Response - Failure:


```
{
  "code": "PAYMENT_INVALID_REQUEST",
  "message": "Date range must not exceed 90 days"
}
```


Error Type Codes

HTTP Status	Error Code	Description
400	BAD_REQUEST	Invalid input value (page/size range, etc.)
401	UNAUTHORIZED	Invalid HMAC
403	FORBIDDEN	Unauthorized IP or Path
503	MAINTENANCE	Under maintenance
406	PAYMENT_INVALID_REQUEST	Missing from/to (when neither transaction nor settlement id is specified), or query period constraint violation
500	INTERNAL_SERVER_ERROR	Internal server error

2.7 Settlement Transaction API (cursor)

The keyset (cursor) variant of the section 2.6 Settlement Transaction API (offset). Use it to traverse the full set, for example to export a CSV. Single-record filters by transactionId and settlementId are not supported.

This API is available only on the v2 path

```
GET /api/seller/v2/payment/settlement/transaction/cursor
```

What to sign: The URI used for the HMAC signature includes the path only, not the query string. See section 2.1 Authentication for details.

Request Query Parameters

Parameter	Type	Required	Description
from	STRING	Required	Query start date (based on capturedAt, ISO 8601 UTC. e.g., 2026-03-01T00:00:00Z). Required, and must be earlier than 'to'.
to	STRING	Required	Query end date (based on capturedAt, ISO 8601 UTC. e.g., 2026-03-31T23:59:59Z). Required; the range is up to 90 days, and it must be at least 1 hour before the current time.
`paymentType`	STRING	Optional	Payment type filter (PURCHASE / REFUND). All types if not specified.
cursor	STRING	Optional	Pass the nextCursor from the previous response as-is. If omitted, returns the first page (opaque base64 token).
size	INTEGER	Optional	Page size (default: 50, max: 100)
order	STRING	Optional	Sort direction DESC (default) / ASC, based on capturedAt.

Example Request:

```
GET /api/seller/v2/payment/settlement/transaction/cursor
    ?from=2026-08-01T00:00:00Z
    &to=2026-08-24T23:59:59Z
    &paymentType=PURCHASE
    &size=50
    &order=DESC
```

Response Fields (200 OK)

Field Name	Type	Required	Description
`content`	List	Required	List of payment records. The item fields are the same as the content of the section 2.6 Settlement Transaction API, and because this endpoint is v2 only, netAmount and protocolFeeRate are always included.
`nextCursor`	STRING	Optional	Next page cursor. Pass it as the cursor in the next request. null on the last page.
`hasNext`	BOOLEAN	Required	Whether a next page exists

`totalElements` / `totalPages` / `pageNumber` / `first` / `last` are not provided (to eliminate count query cost).

Example Response - Success:


```
{
  "content": [
    {
      "partnerCorpName": "Example Corp",
      "storeId": "123",
      "orderId": "LINK-3kQ9xR2mN7pLw8vBc1dEfH-a1B2c3D4e5F6",
      "itemName": "Premium Package",
      "serviceName": "GameShop",
      "orderCountryCode": "KOR",
      "orderCurrencyCode": "USD",
      "orderAmount": 100.0,
      "createdAt": "2026-08-20T10:04:50.000Z",
      "transactionId": "550e8400-e29b-41d4-a716-446655440000",
      "blockchainTxId": "0x7ad1c93ff2e4b8a0cf58d3011992abc...",
      "paymentType": "PURCHASE",
      "originTransactionId": null,
      "status": "CONFIRMED",
      "finalizedAt": "2026-08-20T10:05:32.000Z",
      "capturedAt": "2026-08-20T10:05:30.000Z",
      "failType": null,
      "payCurrencyCode": "USDT",
      "payAmount": 99.95,
      "netAmount": 98.95,
      "buyerWalletAddress": "0xabc...123",
      "paymentExchangeRate": 1.0,
      "variableFeeRate": 0,
      "protocolFeeRate": 1,
      "settlementId": "SETTLE-20260821-004",
      "settlementCurrencyCode": "USDT",
      "settlementExchangeRate": 1.0
    }
  ],
  "nextCursor":
"MTc3MjM1OTUzMDAwMDo1NTBlODQwMC1lMjliLTQxZDQtYTcxNi00NDY2NTU0NDAwMDA",
  "hasNext": true
}
```


Cursor Token: The cursor is an opaque token encoded as URL-safe base64 based on the sort keys (capturedAt, transactionId) - not encryption. Partners simply pass the

nextCursor from the response into the next request as-is, and do not generate or manipulate it directly.

Differences from the offset method

Item	2.6 offset	2.7 cursor
Page navigation	page / size	cursor / size
Total count	totalElements provided	Not provided
Single-record filter	transactionId / settlementId supported	Not supported
Large-volume performance	Degrades at deep offsets	Constant (O(size))
Intended use	Screen pagination	File download and bulk traversal

Error Type Codes

HTTP Status	Error Code	Description
400	BAD_REQUEST	Invalid input value
401	UNAUTHORIZED	Invalid HMAC
403	FORBIDDEN	Unauthorized IP or Path
406	PAYMENT_INVALID_REQUEST	Invalid date format, from or to missing, the range exceeds 90 days, or to is within 1 hour of now
429	RATE_LIMIT_EXCEEDED	More than 10 requests per second. Each settlement query endpoint is limited with its own bucket.
500	INTERNAL_SERVER_ERROR	Internal server error
503	MAINTENANCE	Under maintenance

2.8 Payment Link List API

Looks up the payment links you have issued, by status. It is the lookup counterpart of the section 2.2 Create Payment Link API.

This API is available only on the v2 path

```
GET /api/seller/v2/payment/link
```

What to sign: The URI used for the HMAC signature includes the path only, not the query string. See section 2.1 Authentication for details.

Request Query Parameters

Parameter	Type	Required	Description
status	STRING	Optional	Status of the links to look up (ACTIVE / EXPIRED). Defaults to ACTIVE.
page	INTEGER	Optional	Page number (starting from 0, default: 0)
size	INTEGER	Optional	Page size (default 20, from 1 to 100)

Note on the default: The default size for this API is 20, which differs from the default of 50 used by the other lookup APIs.

Example Request:

```
GET /api/seller/v2/payment/link
    ?status=ACTIVE
    &page=0
    &size=20
```

Response Fields (200 OK)

Field Name	Type	Required	Description
`content`	List	Required	List of payment links.
`content[].linkId`	STRING	Required	Link identifier.
`content[].linkUrl`	STRING	Required	Payment link URL.
`content[].storeId`	STRING	Required	Partner's internal store ID.
`content[].serviceName`	STRING	Required	Service name.
`content[].itemName`	STRING	Required	Item name.
`content[].itemPrice`	DOUBLE	Required	Item price.
`content[].orderCurrencyCode`	STRING	Required	Order currency code.
`content[].status`	STRING	Required	Link status (ACTIVE / EXPIRED). Derived from whether expiresAt has passed at the time of the query.
`content[].returnUrl`	STRING	Optional	The value set at issuance. Null when not set.
`content[].callbackUrl`	STRING	Optional	The value set at issuance. Null when not set.
`content[].expiresAt`	INSTANT	Required	Time the link expires.
`content[].createdAt`	INSTANT	Required	Time the link was issued.

Field Name	Type	Required	Description
`totalElements`	LONG	Required	Total number of records
`totalPages`	INTEGER	Required	Total number of pages
`pageNumber`	INTEGER	Required	Current page number
`first`	BOOLEAN	Required	Whether this is the first page
`last`	BOOLEAN	Required	Whether this is the last page

How status is derived: The status in the response is not a stored state; it is derived from whether expiresAt has passed at the time of the query. A forced expiration also moves expiresAt forward, so it appears as EXPIRED just like a natural expiration. A status=EXPIRED query returns only links within 30 days after expiration; older links are excluded from the list.

Example Response - Success:


```
{
  "content": [
    {
      "linkId": "3kQ9xR2mN7pLw8vBc1dEfH",
      "linkUrl": "https://unifipay.example.com/pay/links/3kQ9xR2mN7pLw8vBc1dEfH",
      "storeId": "STORE-001",
      "serviceName": "GameShop",
      "itemName": "Premium Package",
      "itemPrice": 100.0,
      "orderCurrencyCode": "USD",
      "status": "ACTIVE",
      "returnUrl": "https://myshop.com/payment/returnUrl",
      "callbackUrl": "https://myshop.com/payment/callbackUrl",
      "expiresAt": "2026-09-26T00:00:00.000Z",
      "createdAt": "2026-08-25T09:12:44.000Z"
    }
  ],
  "totalElements": 1,
  "totalPages": 1,
  "pageNumber": 0,
  "first": true,
  "last": true
}
```


Error Type Codes

HTTP Status	Error Code	Description
400	BAD_REQUEST	Invalid input value
401	UNAUTHORIZED	Invalid HMAC
403	FORBIDDEN	Unauthorized IP or Path
500	INTERNAL_SERVER_ERROR	Internal server error
503	MAINTENANCE	Under maintenance

2.9 Force Expire Payment Link API

Force expires an issued payment link. It moves expiresAt to the time of the call to block further checkout entry, and it does not affect payments that have already been created.

This API is available only on the v2 path

```
POST /api/seller/v2/payment/link/{linkId}/disable
```

Irreversible: A forced expiration cannot be undone. An expired link cannot be used again, so verify the target link before calling.

Request Body Parameters

Parameter	Type	Required	Description
`linkId`	STRING	Required	Identifier of the link to expire. Use the linkId from the payment link creation response as is.

There is no request body. Pass only the linkId path variable.

Response Fields (200 OK)

Field Name	Type	Required	Description
`linkId`	STRING	Required	Link identifier.
`linkUrl`	STRING	Required	Payment link URL.
`storeId`	STRING	Required	Partner's internal store ID.
`serviceName`	STRING	Required	Service name.
`itemName`	STRING	Required	Item name.
`itemPrice`	DOUBLE	Required	Item price.
`orderCurrencyCode`	STRING	Required	Order currency code.
`status`	STRING	Required	

Field Name	Type	Required	Description
			Because this is the response right after the forced expiration, it is always EXPIRED.
`returnUrl`	STRING	Optional	The value set at issuance. Null when not set.
`callbackUrl`	STRING	Optional	The value set at issuance. Null when not set.
`expiresAt`	INSTANT	Required	Updated to the time of the expiration call.
`createdAt`	INSTANT	Required	Time the link was issued. The original value is kept.

Example Response:

```
{
  "linkId": "3kQ9xR2mN7pLw8vBc1dEfH",
  "linkUrl": "https://unifipay.example.com/pay/links/3kQ9xR2mN7pLw8vBc1dEfH",
  "storeId": "STORE-001",
  "serviceName": "GameShop",
  "itemName": "Premium Package",
  "itemPrice": 100.0,
  "orderCurrencyCode": "USD",
  "status": "EXPIRED",
  "returnUrl": "https://myshop.com/payment/returnUrl",
  "callbackUrl": "https://myshop.com/payment/callbackUrl",
  "expiresAt": "2026-09-07T10:20:00.000Z",
  "createdAt": "2026-08-25T09:12:44.000Z"
}
```

Error Type Codes

HTTP Status	Error Code	Description
406	PAYMENT_LINK_INVALID	The linkId does not exist or the owner does not match. Both cases return the same code so that the existence of a link is not revealed.
401	UNAUTHORIZED	Invalid HMAC
403	FORBIDDEN	Unauthorized IP or Path
500	INTERNAL_SERVER_ERROR	Internal server error
503	MAINTENANCE	Under maintenance

2.10 Payment Link Transaction API

Looks up transactions created from a single payment link, in all statuses. Transactions of an expired link are also returned.

This API is available only on the v2 path

```
GET /api/seller/v2/payment/link/{linkId}/transaction
```

What to sign: The URI used for the HMAC signature includes the path only, not the query string. See section 2.1 Authentication for details.

Request Query Parameters

Parameter	Type	Required	Description
`linkId`	STRING	Required	Link identifier (22 base62 characters). Use the linkId from the payment link creation response as is. A malformed value returns 400.
to	STRING	Required	End of the query range (based on createdAt, ISO 8601 UTC). This point is not included in the range.
from	STRING	Optional	Start of the query range (based on createdAt, ISO 8601 UTC). This point is included in the range. When omitted, the time the link was issued is used.
status	STRING	Optional	Payment status filter. Multiple values can be given, separated by commas. All statuses when not specified.
`paymentType`	STRING	Optional	Payment type filter (PURCHASE / REFUND). All types if not specified.
cursor	STRING	Optional	Pass the nextCursor from the previous response as is. Omit it for the first page. There is no page parameter.
size	INTEGER	Optional	Page size (default 50, from 1 to 100)
order	STRING	Optional	Sort direction, DESC (default) or ASC, based on createdAt.

Range boundaries: The range is from `from` inclusive to `to` exclusive, so `to` is not part of it. When `from` is omitted it is filled with the time the link was issued, and the range can span at most 90 days.

Refunds are included: A refund inherits the orderId of the original payment, so purchases and refunds are returned together when paymentType is not specified.

Example Request:

```
GET /api/seller/v2/payment/link/Qx4nW2GRAEFkZQsdq7Fj1F/transaction
    ?to=2026-08-31T00:00:00Z
```

```
    &from=2026-08-01T00:00:00Z
    &status=CONFIRMED,PENDING
    &paymentType=PURCHASE
    &size=50
    &order=DESC
```

Response Fields (200 OK)

Field Name	Type	Required	Description
`content`	List	Required	List of transactions.
`content[].transactionId`	STRING	Required	Transaction number generated by Unifi Pay.
`content[].orderId`	STRING	Required	Order ID. For transactions created from a payment link, the server issues it as the LINK prefix, the linkId, and a random string joined by hyphens.
`content[].storeId`	STRING	Required	Merchant ID information provided by the requester.
`content[].serviceName`	STRING	Required	Service or merchant information to be displayed on the payment page.
`content[].items`	List	Required	List of purchased items.
`content[].items[].name`	STRING	Required	Identification information for the item.
`content[].items[].price`	DOUBLE	Required	Price of the item.
`content[].paymentType`	STRING	Required	Payment: PURCHASE / Refund: REFUND
`content[].status`	STRING	Required	Payment status (public representation). One of CREATED, PENDING, PAID, CONFIRMED, FAILED, or CANCELED.
`content[].failType`	STRING	Optional	Set when status is FAILED, and null otherwise. The values are WALLET_PURCHASE_BLOCKCHAIN_FAILED, WALLET_REFUND_BLOCKCHAIN_FAILED, WALLET_PURCHASE_INSUFFICIENT_BALANCE, WALLET_REFUND_INSUFFICIENT_BALANCE, PAYMENT_PURCHASE_BLACKLISTED, PAYMENT_STORE_INACTIVE, WALLET_SESSION_EXPIRED, WALLET_TRANSFER_IN_PROGRESS, and UNKNOWN.
`content[].countryCode`	STRING	Required	Country code following ISO Alpha-3 (e.g., KOR).
`content[].orderCurrencyCode`	STRING	Required	Order currency code following ISO 4217
`content[].orderAmount`	DOUBLE	Required	Order amount

Field Name	Type	Required	Description
`content[].payCurrencyCode`	STRING	Optional	Payment token currency (USDT, JPYC). Null before confirmation.
`content[].payAmount`	DOUBLE	Optional	Token amount actually paid. Null before confirmation.
`content[].netAmount`	DOUBLE	Optional	Net amount the partner actually receives, in the payment currency. Provided from CONFIRMED onward.
`content[].buyerWalletAddress`	STRING	Optional	Buyer wallet address. Null before confirmation.
`content[].blockchainTxId`	STRING	Optional	Blockchain transaction hash. Returned after the transaction is confirmed on the blockchain (CONFIRMED), or when it fails (FAILED) after being propagated to the blockchain network.
`content[].blockchainNetworkFee`	DOUBLE	Optional	Blockchain network fee. The amount of fee actually consumed is returned after the transaction is confirmed on the blockchain (CONFIRMED), or upon failure (FAILED) after network propagation. (Fully covered by Unifi, so there is no cost to the user.)
`content[].createdAt`	INSTANT	Required	Time the payment was created (ISO 8601 UTC). It is the basis for the range, the sort order, and the cursor.
`content[].capturedAt`	INSTANT	Optional	Time when the on-chain event was observed. Null before observation. A value here does not mean the payment succeeded, so determine success from status.
`content[].finalizedAt`	INSTANT	Optional	Time when the payment ended (confirmed, failed, or canceled). Null before it ends.
`nextCursor`	STRING	Optional	Next page cursor. Pass it as the cursor in the next request. null on the last page.
`hasNext`	BOOLEAN	Required	Whether a next page exists

Traversal note: The total count is not provided, so keep calling with nextCursor while hasNext is true. Do not change from, to, status, paymentType, or order during the traversal.

Example Response - Success:


```
{
  "content": [
    {
      "transactionId": "3f1d0a54-9c1e-4f6a-8a5b-2f0011223344",
```

```
      "orderId": "LINK-Qx4nW2GRAEFkZQsdq7Fj1F-7bK2mQ9xR4pL",
      "storeId": "123",
      "serviceName": "GameShop",
      "items": [
        {
          "name": "ITEM-GAME-001",
          "price": 50.0
        }
      ],
      "paymentType": "PURCHASE",
      "status": "CONFIRMED",
      "failType": null,
      "countryCode": "KOR",
      "orderCurrencyCode": "USD",
      "orderAmount": 50.0,
      "payCurrencyCode": "USDT",
      "payAmount": 49.98,
      "netAmount": 49.48,
      "buyerWalletAddress": "0xabc...123",
      "blockchainTxId": "0x4fce0d72ff7f4b9f4ba0cf58d3011992...",
      "blockchainNetworkFee": 0.00253778,
      "createdAt": "2026-08-01T04:00:00.000Z",
      "capturedAt": "2026-08-01T04:01:28.000Z",
      "finalizedAt": "2026-08-01T04:01:30.000Z"
    }
  ],
  "nextCursor": "eyJ0cyI6IjIwMjYtMDgtMDFUMDQ6MDA6MDBaIiwidHgiOiIzZjFkMGE1NCJ9",
  "hasNext": true
}
```


Error Type Codes

HTTP Status	Error Code	Description
400	BAD_REQUEST	Malformed linkId (not 22 base62 characters), size out of range, and so on
406	PAYMENT_LINK_INVALID	The linkId does not exist or the owner does not match. Both cases return the same code so that the existence of a link is not revealed.
406	PAYMENT_INVALID_REQUEST	Invalid date format, from later than to, or a range exceeding 90 days. When from is omitted, the message includes the time the link was issued.
401	UNAUTHORIZED	Invalid HMAC

HTTP Status	Error Code	Description
403	FORBIDDEN	Unauthorized IP or Path
500	INTERNAL_SERVER_ERROR	Internal server error
503	MAINTENANCE	Under maintenance

2.11 Payment Transaction API

Looks up payments within a period, in all statuses. Unlike the section 2.6 Settlement Transaction API, which returns only settlement targets (CONFIRMED), this API also includes CREATED, PENDING, FAILED, and CANCELED.

This API is available only on the v2 path

```
GET /api/seller/v2/payment/transaction
```

What to sign: The URI used for the HMAC signature includes the path only, not the query string. See section 2.1 Authentication for details.

Request Query Parameters

Parameter	Type	Required	Description
from	STRING	Required	Start of the query range (based on createdAt, ISO 8601 UTC). This point is included, and the range can span at most 90 days.
to	STRING	Required	End of the query range (based on createdAt, ISO 8601 UTC). This point is not included in the range.
status	STRING	Optional	Payment status filter. Multiple values can be given, separated by commas. All statuses when not specified.
`paymentType`	STRING	Optional	Payment type filter (PURCHASE / REFUND). All types if not specified.
cursor	STRING	Optional	Pass the nextCursor from the previous response as is. Omit it for the first page. There is no page parameter.
size	INTEGER	Optional	Page size (default 50, from 1 to 100)
order	STRING	Optional	Sort direction, DESC (default) or ASC, based on createdAt.

Example Request:

```
GET /api/seller/v2/payment/transaction
    ?from=2026-08-01T00:00:00Z
```

```
    &to=2026-08-31T00:00:00Z
    &status=CONFIRMED,PENDING
    &paymentType=PURCHASE
    &size=50
    &order=DESC
```

Response Fields (200 OK)

Field Name	Type	Required	Description
`content`	List	Required	List of transactions.
`content[].transactionId`	STRING	Required	Transaction number generated by Unifi Pay.
`content[].orderId`	STRING	Required	Order ID. For transactions created from a payment link, the server issues it as the LINK prefix, the linkId, and a random string joined by hyphens.
`content[].storeId`	STRING	Required	Merchant ID information provided by the requester.
`content[].serviceName`	STRING	Required	Service or merchant information to be displayed on the payment page.
`content[].items`	List	Required	List of purchased items.
`content[].items[].name`	STRING	Required	Identification information for the item.
`content[].items[].price`	DOUBLE	Required	Price of the item.
`content[].paymentType`	STRING	Required	Payment: PURCHASE / Refund: REFUND
`content[].status`	STRING	Required	Payment status (public representation). One of CREATED, PENDING, PAID, CONFIRMED, FAILED, or CANCELED.
`content[].failType`	STRING	Optional	Set when status is FAILED, and null otherwise. The values are WALLET_PURCHASE_BLOCKCHAIN_FAILED, WALLET_REFUND_BLOCKCHAIN_FAILED, WALLET_PURCHASE_INSUFFICIENT_BALANCE, WALLET_REFUND_INSUFFICIENT_BALANCE, PAYMENT_PURCHASE_BLACKLISTED, PAYMENT_STORE_INACTIVE, WALLET_SESSION_EXPIRED, WALLET_TRANSFER_IN_PROGRESS, and UNKNOWN.
`content[].countryCode`	STRING	Required	Country code following ISO Alpha-3 (e.g., KOR).
`content[].orderCurrencyCode`	STRING	Required	Order currency code following ISO 4217
`content[].orderAmount`	DOUBLE	Required	Order amount

Field Name	Type	Required	Description
`content[].payCurrencyCode`	STRING	Optional	Payment token currency (USDT, JPYC). Null before confirmation.
`content[].payAmount`	DOUBLE	Optional	Token amount actually paid. Null before confirmation.
`content[].netAmount`	DOUBLE	Optional	Net amount the partner actually receives, in the payment currency. Provided from CONFIRMED onward.
`content[].buyerWalletAddress`	STRING	Optional	Buyer wallet address. Null before confirmation.
`content[].blockchainTxId`	STRING	Optional	Blockchain transaction hash. Returned after the transaction is confirmed on the blockchain (CONFIRMED), or when it fails (FAILED) after being propagated to the blockchain network.
`content[].blockchainNetworkFee`	DOUBLE	Optional	Blockchain network fee. The amount of fee actually consumed is returned after the transaction is confirmed on the blockchain (CONFIRMED), or upon failure (FAILED) after network propagation. (Fully covered by Unifi, so there is no cost to the user.)
`content[].createdAt`	INSTANT	Required	Time the payment was created (ISO 8601 UTC). It is the basis for the range, the sort order, and the cursor.
`content[].capturedAt`	INSTANT	Optional	Time when the on-chain event was observed. Null before observation. A value here does not mean the payment succeeded, so determine success from status.
`content[].finalizedAt`	INSTANT	Optional	Time when the payment ended (confirmed, failed, or canceled). Null before it ends.
`nextCursor`	STRING	Optional	Next page cursor. Pass it as the cursor in the next request. null on the last page.
`hasNext`	BOOLEAN	Required	Whether a next page exists

Traversal note: The total count is not provided, so keep calling with nextCursor while hasNext is true. Do not change from, to, status, paymentType, or order during the traversal.

Note on mixing versions: A payment created with v1 that is in PAID matches the status=PAID filter, but the status in the response comes back as CONFIRMED. PAID is exposed only when both the version that created the payment and the version used to query are v2.

Example Response - Success:

```
{
  "content": [
    {
      "transactionId": "3f1d0a54-9c1e-4f6a-8a5b-2f0011223344",
      "orderId": "ORD-20260304-0001",
      "storeId": "123",
      "serviceName": "GameShop",
      "items": [
        {
          "name": "ITEM-GAME-001",
          "price": 50.0
        }
      ],
      "paymentType": "PURCHASE",
      "status": "CONFIRMED",
      "failType": null,
      "countryCode": "KOR",
      "orderCurrencyCode": "USD",
      "orderAmount": 50.0,
      "payCurrencyCode": "USDT",
      "payAmount": 49.98,
      "netAmount": 49.48,
      "buyerWalletAddress": "0xabc...123",
      "blockchainTxId": "0x4fce0d72ff7f4b9f4ba0cf58d3011992...",
      "blockchainNetworkFee": 0.00253778,
      "createdAt": "2026-08-01T04:00:00.000Z",
      "capturedAt": "2026-08-01T04:01:28.000Z",
      "finalizedAt": "2026-08-01T04:01:30.000Z"
    }
  ],
  "nextCursor": "eyJ0cyI6IjIwMjYtMDgtMDFUMDQ6MDA6MDBaIiwidHgiOiIzZjFkMGE1NCJ9",
  "hasNext": true
}
```


Error Type Codes

HTTP Status	Error Code	Description
400	BAD_REQUEST	from or to missing, size out of range, an unknown status, paymentType, or order value, and so on
406	PAYMENT_INVALID_REQUEST	Invalid date format (not ISO 8601), from later than to, or a range exceeding 90 days

HTTP Status	Error Code	Description
401	UNAUTHORIZED	Invalid HMAC
403	FORBIDDEN	Unauthorized IP or Path
500	INTERNAL_SERVER_ERROR	Internal server error
503	MAINTENANCE	Under maintenance

2.12 Store List API

Looks up the stores that belong to your partner account. A payment link belongs to a store, and the order currency and payment currency follow that store, so use this API to check the partnerStoreId and the allowed currencies before calling the section 2.2 Create Payment Link API.

This API is available only on the v2 path

```
GET /api/seller/v2/store
```

What to sign: The URI used for the HMAC signature includes the path only, not the query string. See section 2.1 Authentication for details.

Request Query Parameters

Parameter	Type	Required	Description
`partnerStoreId`	STRING	Optional	Filters by the partner's internal store ID. All stores when not specified.
status	STRING	Optional	Status of the stores to look up (ACTIVE / INACTIVE). All stores when not specified.
page	INTEGER	Optional	Page number (starting from 0, default: 0)
size	INTEGER	Optional	Page size (default 20, from 1 to 50)

Example Request:

```
GET /api/seller/v2/store
    ?status=ACTIVE
    &page=0
    &size=20
```

Response Fields (200 OK)

Field Name	Type	Required	Description
`content`	List	Required	List of stores.
`content[].partnerStoreId`	STRING	Required	The partner's internal store ID. This is the value used as storeId when creating a payment link.
`content[].name`	STRING	Required	Store name.
`content[].status`	STRING	Required	Store status (ACTIVE / INACTIVE).
`content[].orderCurrencyCodes`	List	Optional	List of allowed order currencies. Null means the partner policy is inherited.
`content[].paymentCurrencyCodes`	List	Optional	List of allowed payment coins. Null means the partner policy is inherited.
`totalElements`	LONG	Required	Total number of records
`totalPages`	INTEGER	Required	Total number of pages
`pageNumber`	INTEGER	Required	Current page number
`first`	BOOLEAN	Required	Whether this is the first page
`last`	BOOLEAN	Required	Whether this is the last page

Currency policy: A store currency is chosen from the partner currencies. When a store currency list is empty or null, the partner policy is inherited as is, and a store currency is always a subset of the partner currencies. If the order currency requested at payment link creation is outside the allowed list, creation is rejected.

Example Response - Success:


```
{
  "content": [
    {
      "partnerStoreId": "STORE-001",
      "name": "GameShop Main",
      "status": "ACTIVE",
      "orderCurrencyCodes": [
        "USD",
        "JPY"
      ],
      "paymentCurrencyCodes": [
        "USDT"
      ]
    }
  ],
  "totalElements": 1,
```

```
  "totalPages": 1,
  "pageNumber": 0,
  "first": true,
  "last": true
}
```

Error Type Codes

HTTP Status	Error Code	Description
400	BAD_REQUEST	Invalid input value
401	UNAUTHORIZED	Invalid HMAC
403	FORBIDDEN	Unauthorized IP or Path
500	INTERNAL_SERVER_ERROR	Internal server error
503	MAINTENANCE	Under maintenance

2.13 Blacklist API

Six APIs that register, list, rescope, and unblock blocked wallets within your own partner scope. Authentication uses the same HMAC scheme as section 2.1 Authentication, and every call operates only within your own partner scope.

This API is available only on the v2 path

Caution: Once a wallet is blocked, its payments or refunds are rejected immediately. Be sure to verify the wallet address, reason, and scope before calling.

Blocks registered by a Unifi Pay operator for the entire scope do not appear in the list or history of this API. Even after a merchant unblocks its own entry, the wallet may still be rejected while an operator block remains.

Wallet Address Format

Item	Description
`walletAddress`	An EVM address of 40 hexadecimal characters starting with 0x. Both uppercase and lowercase are allowed (^0x[0-9a-fA-F]{40}$).

Block Reason (reason)

Value	Description
`FRAUD_PATTERN`	Repeated abnormal transactions
`REFUND_ABUSE`	Abnormal refund requests

Value	Description
`AML_SUSPECT`	Suspected money laundering
`BLACK_CONSUMER`	Black consumer
`OTHER`	Other. Supplement it with memo.

Block Type (blockTypes)

Value	Description
`PURCHASE`	Blocks payments.
`REFUND`	Blocks refunds.

Block scope: scopeType has two values, PARTNER (the whole partner) and STORE (the specified stores). The register request body has no scopeType field: leaving storeIds empty resolves to PARTNER, and specifying stores resolves to STORE.

Failure codes for a blocked wallet: A payment from a blocked wallet fails with PAYMENT_PURCHASE_BLACKLISTED, listed in the failType table of section 2.3 Payment Status API. A refund is rejected with PAYMENT_REFUND_BLACKLISTED, listed in the refund request error table of section 2.5 Refund API (DIRECT).

2.13.1 Blocked Wallet List API

Looks up the blocked wallets registered within your own partner scope. The id in the response is used as the path variable for changing the block scope and for unblocking.

```
GET /api/seller/v2/blacklist/search
```

Request Query Parameters

Parameter	Type	Required	Description
`walletAddress`	STRING	Optional	Filters by wallet address.
`reason`	STRING	Optional	Filters by block reason (FRAUD_PATTERN / REFUND_ABUSE / AML_SUSPECT / BLACK_CONSUMER / OTHER). All reasons when not specified.
`scopeType`	STRING	Optional	Filters by block scope (PARTNER / STORE). All scopes when not specified.
`blockType`	STRING	Optional	Filters by block type (PURCHASE / REFUND). All types when not specified.
`page`	INTEGER	Optional	Page number (starting from 0, default: 0)
`size`	INTEGER	Optional	Page size (default 20, from 1 to 100)

Example Request:

```
GET /api/seller/v2/blacklist/search
    ?scopeType=PARTNER
    &page=0
    &size=20
```

Response Fields (200 OK)

Field Name	Type	Required	Description
`content`	List	Required	List of blocked wallets.
`content[].id`	LONG	Required	Block entry ID. Used as the {id} path variable for changing the block scope and for unblocking.
`content[].walletAddress`	STRING	Required	The blocked wallet address.
`content[].reason`	STRING	Required	Block reason.
`content[].blockTypes`	List	Optional	Block types (PURCHASE / REFUND).
`content[].scopeType`	STRING	Required	Block scope (PARTNER / STORE).
`content[].scopes`	List	Required	List of applied scopes (partnerId, storeIds).
`content[].active`	BOOLEAN	Required	Whether the entry is active.
`content[].memo`	STRING	Optional	Memo.
`content[].registeredBy`	STRING	Optional	The registrant (appId).
`content[].registeredAt`	INSTANT	Required	Registration time.
`totalElements`	LONG	Required	Total number of records
`totalPages`	INTEGER	Required	Total number of pages
`pageNumber`	INTEGER	Required	Current page number
`first`	BOOLEAN	Required	Whether this is the first page
`last`	BOOLEAN	Required	Whether this is the last page

Example Response - Success:


```
{
  "content": [
    {
      "id": 3012,
      "walletAddress": "0x1234567890abcdef1234567890abcdef12345678",
      "reason": "REFUND_ABUSE",
```

```
      "blockTypes": [
        "PURCHASE",
        "REFUND"
      ],
      "scopeType": "PARTNER",
      "scopes": [
        {
          "partnerId": "55",
          "storeIds": null
        }
      ],
      "active": true,
      "memo": "Repeated refund requests",
      "registeredBy": "seller-app-001",
      "registeredAt": "2026-08-24T05:31:00.000Z"
    }
  ],
  "totalElements": 1,
  "totalPages": 1,
  "pageNumber": 0,
  "first": true,
  "last": true
}
```


Error Type Codes

HTTP Status	Error Code	Description
400	BAD_REQUEST	Invalid input value
401	UNAUTHORIZED	Invalid HMAC
403	FORBIDDEN	Unauthorized IP or Path
500	INTERNAL_SERVER_ERROR	Internal server error
503	MAINTENANCE	Under maintenance

2.13.2 Register Block API

Registers a wallet in the block list. There is no API for changing the reason or the memo, so unblock the entry and register it again; only the scope can be changed with the Change Block Scope API.

Caution: Once a wallet is blocked, its payments or refunds are rejected immediately. Be sure to verify the wallet address, reason, and scope before calling.

```
POST /api/seller/v2/blacklist
```

Request Body Parameters

Parameter	Type	Required	Description
`walletAddress`	STRING	Required	The wallet address to block. Enter an EVM address of 40 hexadecimal characters starting with 0x.
`reason`	STRING	Required	The block reason (FRAUD_PATTERN / REFUND_ABUSE / AML_SUSPECT / BLACK_CONSUMER / OTHER).
`blockTypes`	List	Optional	The block types (PURCHASE / REFUND). All types are blocked when omitted.
`storeIds`	List	Optional	Leave it empty to block the whole partner, or specify stores to block only those stores. Up to 50 entries, each up to 64 characters. 0 is a reserved value and cannot be used.
`memo`	STRING	Optional	A memo. Up to 1,000 characters.

Example Request:

```
{
  "walletAddress": "0x1234567890abcdef1234567890abcdef12345678",
  "reason": "REFUND_ABUSE",
  "blockTypes": [
    "PURCHASE",
    "REFUND"
  ],
  "storeIds": [
    "STORE-001"
  ],
  "memo": "Repeated refund requests"
}
```

Response Fields (200 OK)

Field Name	Type	Required	Description
id	LONG	Required	The ID of the registered block entry.

Example Response - Success:

```
{
  "id": 3012
}
```

Error Type Codes

HTTP Status	Error Code	Description
400	BAD_REQUEST	Invalid wallet address format, or storeIds contains the reserved value 0
401	UNAUTHORIZED	Invalid HMAC
403	FORBIDDEN	Unauthorized IP or Path
406	BLACKLIST_ALREADY_EXIST	The wallet is already blocked. An active entry exists for the same partner.
500	INTERNAL_SERVER_ERROR	Internal server error
503	MAINTENANCE	Under maintenance

2.13.3 Change Block Scope API

Switches the applied scope of a block between the whole partner and the specified stores. The {id} path variable is the id from the Blocked Wallet List API response.

```
PUT /api/seller/v2/blacklist/{id}/scope
```

Request Body Parameters

Parameter	Type	Required	Description
`storeIds`	List	Optional	Leave it empty to block the whole partner, or specify stores to block only those stores. Up to 50 entries, each up to 64 characters. 0 is a reserved value and cannot be used.

Full replacement: storeIds replaces the entire existing scope rather than partially editing it. Omitting the field or sending null switches the block to the whole partner, while sending an empty array is rejected with 400.

Example Request:

```
{
  "storeIds": [
    "STORE-001"
  ]
}
```

Response Fields (200 OK): The 200 OK response has no body.

Error Type Codes

HTTP Status	Error Code	Description
400	BAD_REQUEST	storeIds is an empty array or contains the reserved value 0
401	UNAUTHORIZED	Invalid HMAC
403	FORBIDDEN	Unauthorized IP or Path
406	PAYMENT_TRANSACTION_NOT_FOUND	The id does not exist or belongs to another partner. Both cases return the same code so that the existence of an entry is not revealed.
500	INTERNAL_SERVER_ERROR	Internal server error
503	MAINTENANCE	Under maintenance

2.13.4 Unblock API

Unblocks an entry you registered. The {id} path variable is the id from the Blocked Wallet List API response.

Caution: The blocked wallet becomes able to pay again. This requires the same level of verification as registering a block.

```
POST /api/seller/v2/blacklist/{id}/unblock
```

Request Body Parameters

Parameter	Type	Required	Description
reason	STRING	Optional	The reason for unblocking. It is recorded in the history and can be up to 1,000 characters.

Example Request:

```
{
  "reason": "Block released after the appeal was approved"
}
```

Response Fields (200 OK): The 200 OK response has no body.

Error Type Codes

HTTP Status	Error Code	Description
401	UNAUTHORIZED	Invalid HMAC
403	FORBIDDEN	Unauthorized IP or Path

HTTP Status	Error Code	Description
406	PAYMENT_TRANSACTION_NOT_FOUND	The id does not exist, has already been unblocked, or belongs to another partner. All cases return the same code so that the existence of an entry is not revealed.
500	INTERNAL_SERVER_ERROR	Internal server error
503	MAINTENANCE	Under maintenance

2.13.5 Block Summary API

Looks up a summary of the number of active blocked wallets and the blocked attempts in the last 24 hours. There are no request parameters.

```
GET /api/seller/v2/blacklist/summary
```

Response Fields (200 OK)

Field Name	Type	Required	Description
`blacklistWalletCount`	LONG	Required	Number of active blocked wallets.
`blockedAttempt.count`	LONG	Required	Number of rejections.
`blockedAttempt.period`	STRING	Optional	Notation for the aggregation period. It is currently fixed to 24H.

Example Response - Success:

```
{
  "blacklistWalletCount": 7,
  "blockedAttempt": {
    "count": 23,
    "period": "24H"
  }
}
```

Error Type Codes

HTTP Status	Error Code	Description
401	UNAUTHORIZED	Invalid HMAC
403	FORBIDDEN	Unauthorized IP or Path
500	INTERNAL_SERVER_ERROR	Internal server error

HTTP Status	Error Code	Description
503	MAINTENANCE	Under maintenance

2.13.6 Block Change History API

Looks up the registration, scope change, and unblock history that affected your own partner. Only paging is provided, with no filters.

```
GET /api/seller/v2/blacklist/history/search
```

Request Query Parameters

Parameter	Type	Required	Description
page	INTEGER	Optional	Page number (starting from 0, default: 0)
size	INTEGER	Optional	Page size (default 20, from 1 to 100)

Example Request:

```
GET /api/seller/v2/blacklist/history/search
    ?page=0
    &size=20
```

Response Fields (200 OK)

Field Name	Type	Required	Description
`content`	List	Required	List of block change history entries.
`content[].id`	LONG	Required	History entry ID.
`content[].blacklistId`	LONG	Required	ID of the target block entry.
`content[].walletAddress`	STRING	Required	Wallet address.
`content[].changeType`	STRING	Required	Change type (REGISTER / SCOPE_CHANGE / UNBLOCK).
`content[].beforeScopes`	List	Optional	Scope before the change (partnerId, storeIds).
`content[].afterScopes`	List	Optional	Scope after the change.
`content[].blockTypes`	List	Optional	Block types.
`content[].unblockReason`	STRING	Optional	Unblock reason (when UNBLOCK).
`content[].createdAt`	INSTANT	Required	Change time.
`totalElements`	LONG	Required	Total number of records

Field Name	Type	Required	Description
`totalPages`	INTEGER	Required	Total number of pages
`pageNumber`	INTEGER	Required	Current page number
`first`	BOOLEAN	Required	Whether this is the first page
`last`	BOOLEAN	Required	Whether this is the last page

Example Response - Success:


```
{
  "content": [
    {
      "id": 9001,
      "blacklistId": 3012,
      "walletAddress": "0x1234567890abcdef1234567890abcdef12345678",
      "changeType": "REGISTER",
      "beforeScopes": null,
      "afterScopes": [
        {
          "partnerId": "55",
          "storeIds": null
        }
      ],
      "blockTypes": [
        "PURCHASE",
        "REFUND"
      ],
      "unblockReason": null,
      "createdAt": "2026-08-24T05:31:00.000Z"
    }
  ],
  "totalElements": 1,
  "totalPages": 1,
  "pageNumber": 0,
  "first": true,
  "last": true
}
```


Error Type Codes

HTTP Status	Error Code	Description
400	BAD_REQUEST	Invalid input value

HTTP Status	Error Code	Description
401	UNAUTHORIZED	Invalid HMAC
403	FORBIDDEN	Unauthorized IP or Path
500	INTERNAL_SERVER_ERROR	Internal server error
503	MAINTENANCE	Under maintenance

---

3. Test Guide

This guide explains how to test Unifi Pay payment integration in the Preview environment.

1. Unifi Pay testing can be conducted in the Preview environment.
2. Unifi Wallets are created and linked in the Preview environment using a Google account that meets Production standards. (Unifi Wallets created in the Preview environment are managed separately from those in the Production environment.)
   Preview Unifi Wallet: https://app.unifi.me/
3. Payments in the Preview environment are processed based on Kairos, the testnet environment of the Kaia blockchain.
4. Follow the guide below to obtain test USDT on the Kairos testnet and then proceed with payment testing.

Test Environment Setup Procedure:

1. **STEP 1 - Create a Unifi Wallet**: Create and link a Unifi Wallet in the Preview environment
2. **STEP 2 - Obtain Test USDT**: Receive Kairos USDT from the Kaia Faucet
3. **STEP 3 - Conduct Payment Testing**: Test API integration in the Preview environment
   Preview Environment Information:

Item	Value
`Environment`	Preview Environment / Kairos Testnet
`Description`	In the Preview environment, the Kairos chain - the testnet for the Kaia blockchain - is integrated, and test funds can be obtained through the Kaia Faucet service.
Kaia Faucet	https://www.kaia.io/ko/faucet
Base URL	https://app-api-pay.unifi.me